Adding Business Readability with SpecFlow

UI AUTOMATION THE BUSINESS UNDERSTANDS

Jason Roberts
.NET MVP

@robertsjason dontcodetired.com

Overview

Why business readable UI automation?

Overview of SpecFlow

Installing SpecFlow in Visual Studio

UI automation styles

Creating business readable SpecFlow scenarios

Adding Selenium automation code

Automation code maintainability

Why Business Readable UI Automation?

Document features

Non-technical

Onboarding

Audit

Executable tests

Source controlled

Stays accurate

“Living”

Better communication

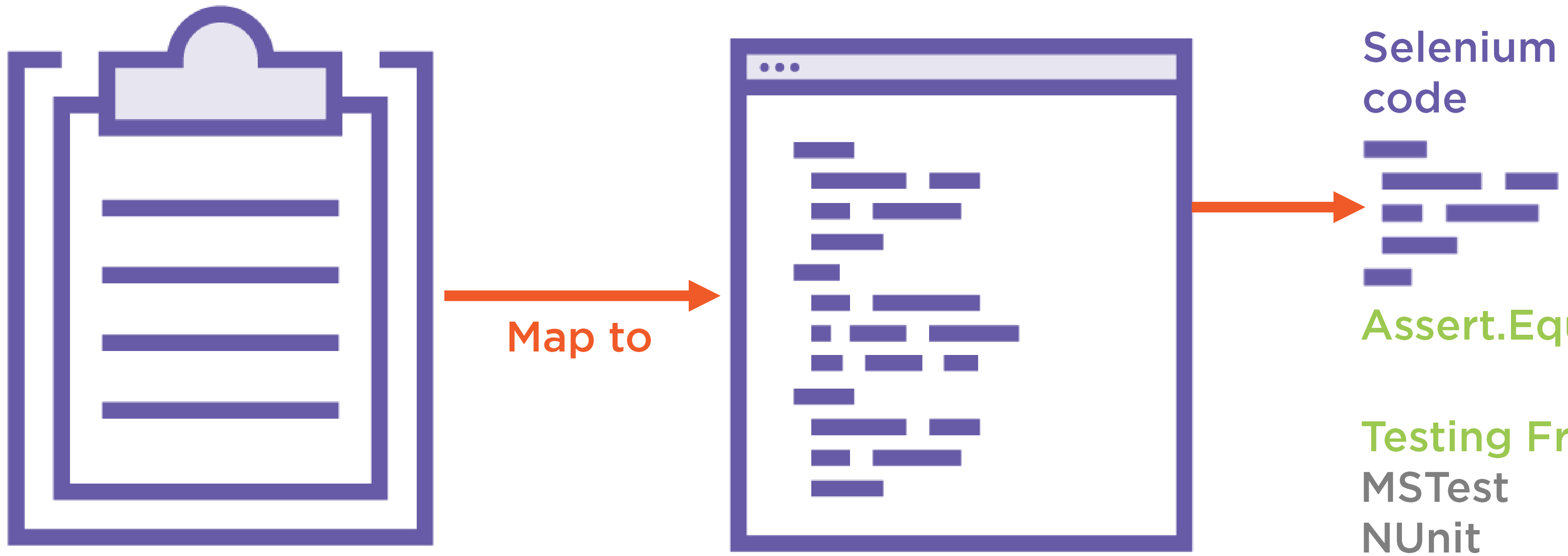
Common/high-level language

Correct features

Reduce wasted effort

“Build the right thing and build it right.”

Overview of SpecFlow



SpecFlow Feature File
(Business readable)

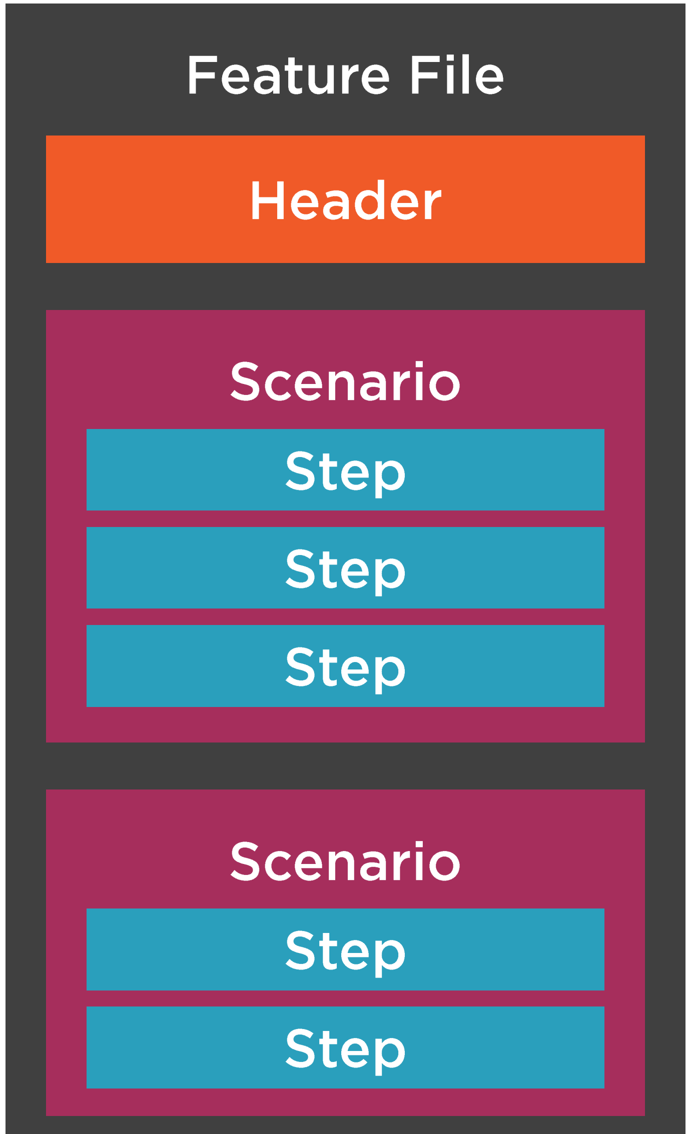
Methods (.cs)

Selenium automation code

Assert.Equal(...)

Testing Framework
MSTest
NUnit
xUnit.net

SpecFlow Feature File Structure



Name and description of feature

Scenario name

Logical steps (high-level / non-technical)

Gherkin Steps

```
Scenario: Application completed successfully
Given I am on the loan application screen
   And I enter a first name of Sarah
   And I enter a second name of Smith
When I submit my application
Then I should see the application complete confirmation
```

Pluralsight Course:

“Business Readable Automated Tests with SpecFlow 2”

Demo

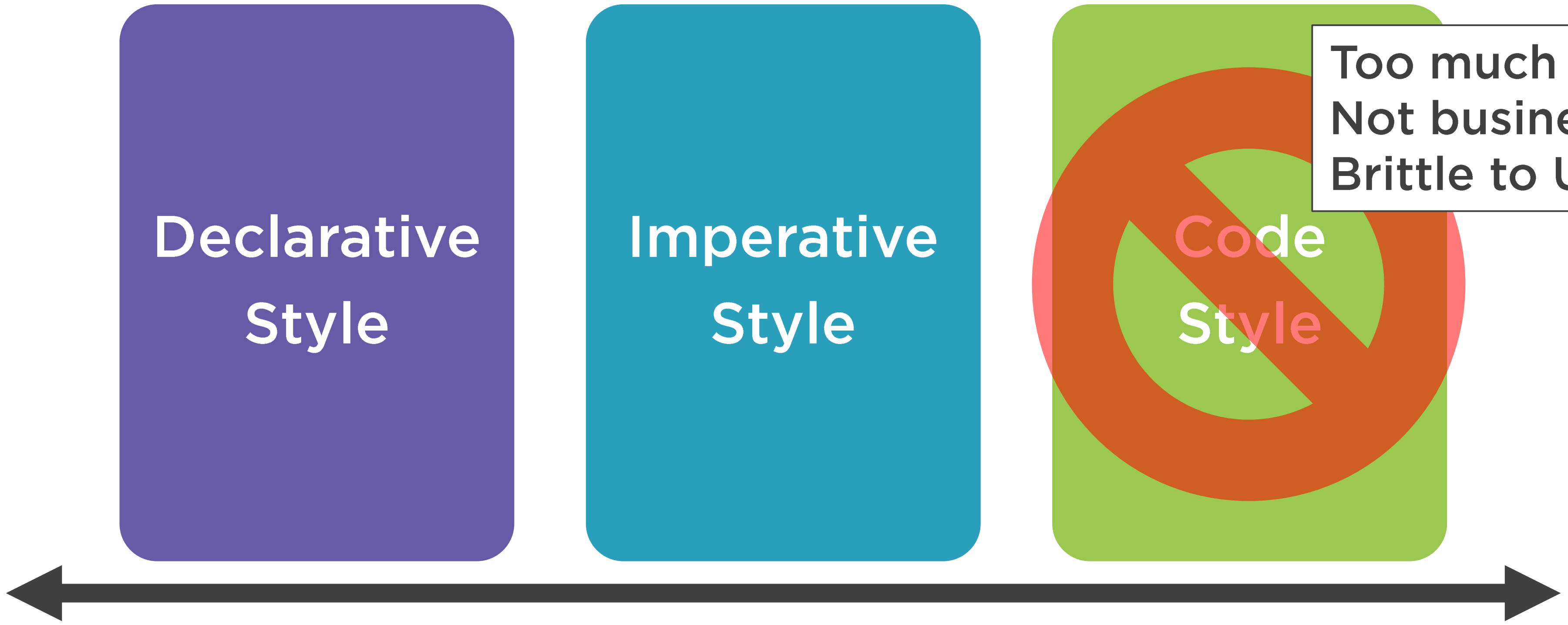
Installing SpecFlow in Visual Studio

Install SpecFlow.xUnit NuGet

Main SpecFlow package + xUnit.net support

SpecFlow Visual Studio Extension

UI Automation Styles



Too much detail
Not business readable
Brittle to UI changes

Less detail
Closer to business
Less technical
Behaviour

More detail
Closer to developers
More technical
Implementation

Code Style

```
Given I navigate to http://localhost:40077/Home/StartLoanApplication
    And I enter Sarah into the input with an ID of FirstName
    And I enter Smith into the input with an ID of LastName
    And I click the element with an ID of Loan
    And I click the element with an ID of TermsAcceptance
When I click the button with a CSS selector of .btn.btn-primary
Then The span with an ID of firstName should contain the text Sarah
```

Code Style

```
Given I navigate to http://localhost:40077/Home/StartLoanApplication
    And I enter Sarah into the input with an ID of FirstName
    And I enter Smith into the input with an ID of LastName
    And I click the element with an ID of Loan
    And I click the element with an ID of TermsAcceptance
When I click the button with a CSS selector of .btn.btn-primary
Then The span with an ID of firstName should contain the text Sarah
```

Imperative Style

UI agnostic

```
Given I am on the loan application screen
    And I enter a first name of Sarah
    And I enter a second name of Smith
    And I select that I have an existing loan account
    And I confirm my acceptance of the terms and conditions
When I submit my application
Then I should see the application complete confirmation for Sarah
```

Declarative Style

UI agnostic
Data agnostic

```
Given I am on the loan application screen
    And I enter valid loan application details
When I submit my application
Then I should see the application complete confirmation
```

When deciding between declarative and imperative styles, talk to the business.

When deciding between declarative and imperative styles, talk to the business.

For some scenarios, the imperative style may enhance their understanding.

Not all scenarios/features have to use same style.

Demo

Adding a SpecFlow Scenario

Add new “LoanApplication” SpecFlow feature file

Add feature description

New scenario: “Application completed successfully”

Write Given, When, and Then steps

Generate step definitions file (containing C# methods)

Demo

Adding the Web Automation Code

Add _driver field

Create FirefoxDriver in Given step definition

Copy Selenium automation code from previous test into Given, When, and Then steps

Dispose in [AfterScenario] SpecFlow hook method

Ensure web application is running

Run scenario

Demo

Creating the Next Scenario

Add new scenario: “Cannot submit application unless terms and conditions accepted”

Don’t want duplicated step definition methods

Modify existing step definitions to create parameterized versions

Generate missing “I should see an error message...” step definition

Add automation code to locate and assert correct error message

Run both scenarios

Automation Code Maintainability Considerations

Reduce duplication

- Element selectors FindElement(...)
- Actions, e.g. clicking specific button

Improve readability

- _driver.FindElement(By.CssSelector(".btn.btn-primary")).Click();
- page.SubmitApplication();

Decrease maintenance cost

- Decouple UI from automation code
- E.g. change element ID in one place

Page Object Models

Summary

Why business readable UI automation?

Executable, living documentation

Easily understood by business people

Overview of SpecFlow

Installed SpecFlow extension and NuGet package in Visual Studio

Code, imperative, and declarative styles

Created SpecFlow scenarios

Generated step definition class file

Added Selenium automation code

Automation code maintainability

Next:

Creating More Maintainable Web Automation